UMTPモデリング技術部会
UML2.5勉強会成果物

UML仕様書の読者向けガイド

1. はじめに

- この資料は、UMTPモデリング技術部会『UML2.5勉強会』の2013年度成果物として作成されました。
  - UMTPについて
    http://www.umtp-japan.org/
  - モデリング技術部会について
    http://www.umtp-japan.org/modules/introduction1/index.php?id=34&tmid0=22
- 当部会2013年度活動予定と結果は以下の通りです。
  - 「モデリング技術の調査・研究及びモデリング技術動向の発信。」
  1. UML規格の最新動向キャッチアップ活動＆成果報告
     - 『UML2.5勉強会』の月例開催
     - 年度末の勉強会成果物(本資料)作成
  2. セミナー&ワークショップによる情報発信
     - モデリングセミナー『DDD（domain-driven design）をじっくり語る60分』開催
       告知URL：http://www.umtp-japan.org/modules/activity2/index.php?id=222

1.1『UML2.5勉強会』について

- 2013年度活動テーマ
  - 「UML規格の最新動向をキャッチアップする」
    - UML 1.xから2.xへの変更点調査
    - 2.xで導入された仕様の理解
    - 2.5での変更点調査
- 変更点調査の結果
  - 意味定義、表記法の変更は無し
  - 仕様書の構成に大きな変更あり

■メンバー（あいうえお順）
河合昭男(有限会社オブジェクトデザイン研究所）
豊田徳子（株式会社オージス総研）
羽生田栄一（株式会社豆蔵）
細谷竜一(株式会社 オージス総研)
山城明宏 (東芝ソリューション株式会社)
吉田裕之(富士通株式会社)
宮川誠（株式会社オージス総研）

仕様変更の内容をまとめるよりも、仕様書を読むにあたってのガイドを作成する方向へ方針を転換した。

1.2 この資料の内容について

- UML仕様は何を定義するものか
  - UML成立の歴史から、標準化の目的を述べる。
  - UMLの各バージョンで導入されたコンセプトを紹介し、定義のバックボーンを知る。
- UML仕様書の読み方
  - UML1.5およびUML2.0仕様書から、UML1.xとUML2.xに共通の部分を明らかにする。
  - UML2.5仕様書の内容を見ながら、その読み方を解説する。
- その他の調査事項

2. UML仕様は何を定義するものか

- UMLの黎明
- The Three Amigos
- UMLの誕生
- UMLの標準化
- UMLの目的
- UMLの成長

2.1 UMLの黎明

1960's	ソフトウェア危機によるソフトウェア工学の発展
1970's	多数の開発方法論が勃興
1980's	**オブジェクト指向言語**の勃興
1990'	設計技術としての**オブジェクト指向**が始まる
1990	GE社の*James Rumbaugh*が**OMT法**を公開
1991	Rational社の*Grady Booch*が**Booch法**を公開
1992	Objectory AB社の*Ivar Jacobson*が**Objectory法**を公開
1994	*Rumbaugh*がRational社へ移籍
1995	Rational社、オブジェクト指向開発手法**Unified Method**発表 *Jacobson*がRational社へ移籍
1996	Rational社、オブジェクト指向開発手法Objectory Process発表 その表記法として、**UML0.9**を発表

2.2 The Three Amigos

UMLの誕生に尽力した偉大な三人のエンジニアと
彼らのオブジェクト指向開発方法論

OMT法
- James Rumbaugh
- 1990年「Object-Oriented Modeling and Design」発表
- 分析に強い

Booch法
- Grady Booch
- 1990年「Object oriented design with applications」出版
- デザインに強い

Objectory(OOSE)法
- Ivar Jacobson
- 1992年「OOSE; Object-Oriented Software Engineering」出版
- ユースケースの導入

初期
- 40種以上の方法論が乱立
- 同様の概念に異なる名称
- 方法論ごとに異なる表記法

過渡期
- 改良、統廃合が進む
- 特定の方法論にモデラーの支持が集中

統一期
- 方法論、記法の統一へ

そして……

2.3 UMLの誕生

✓ Booch法にOMTアプローチを統合

✓ OMT法にBooch記法とユースケースが導入される

✓ OOPSLA 95'でUnified Method発表

✓ UMLにObjectory法を導入

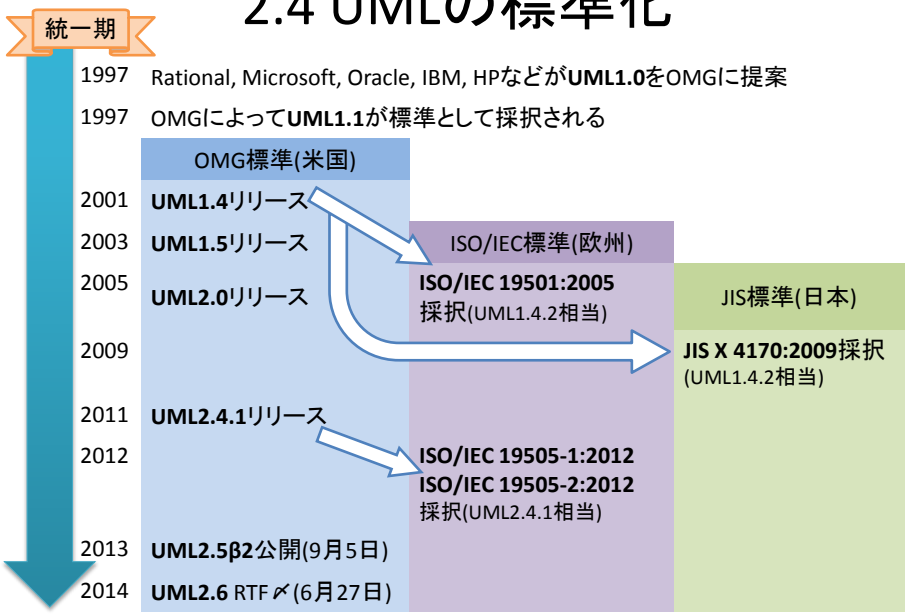
プロセスと言語の仕様を分離

言語→Unified Modeling Language(UML)

プロセス→Rational Objectory Process(後のRUP)

イラストは https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/truthaboutagile?lang=en より引用

2.4 UMLの標準化



2.5 UMLが作られた目的

モデルの視覚化
- 特定の開発方法論に依存することなく、モデルを視覚化すること

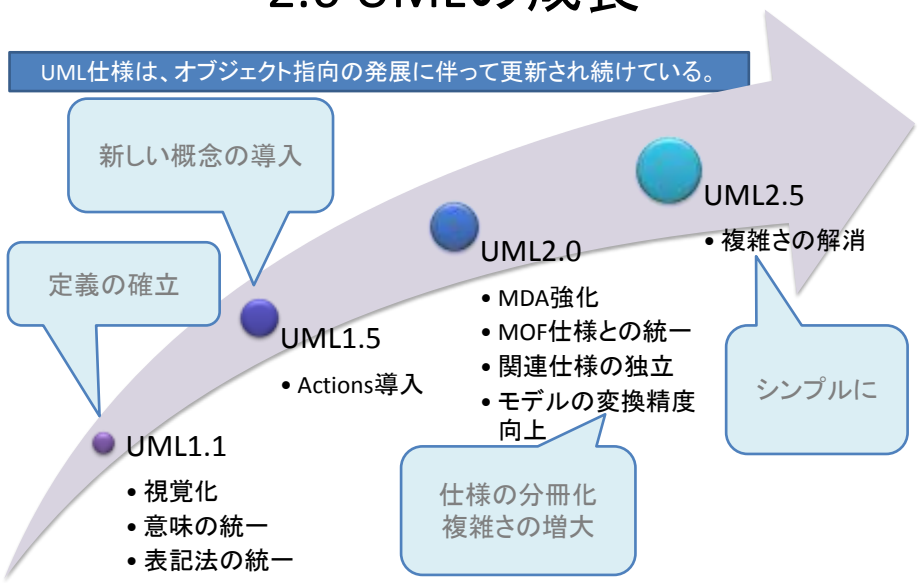
モデルの相互理解
- 開発関係者間で誤解や論争が発生しないように、相互理解を可能にすること

多様なシステム範囲への適用
- さまざまな視点からのモデリングをサポートすること

モデルの意味統一
- モデリングに使用する要素(クラスや関連など)の定義を統一すること

モデルの表記法統一
- 要素の表記法を統一すること

2.6 UMLの成長



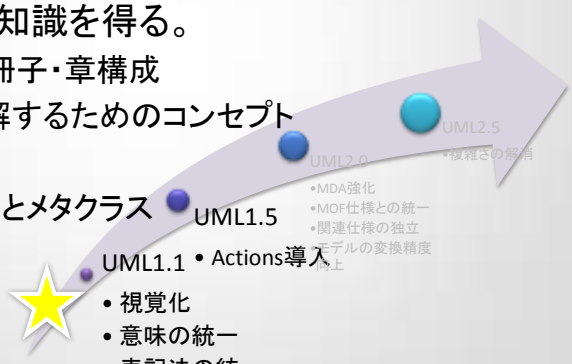
3. UML仕様書の読み方

- UML1.5仕様書の構造
- UML2.0仕様書の構造
- UML2.5での変更点

UML1.5

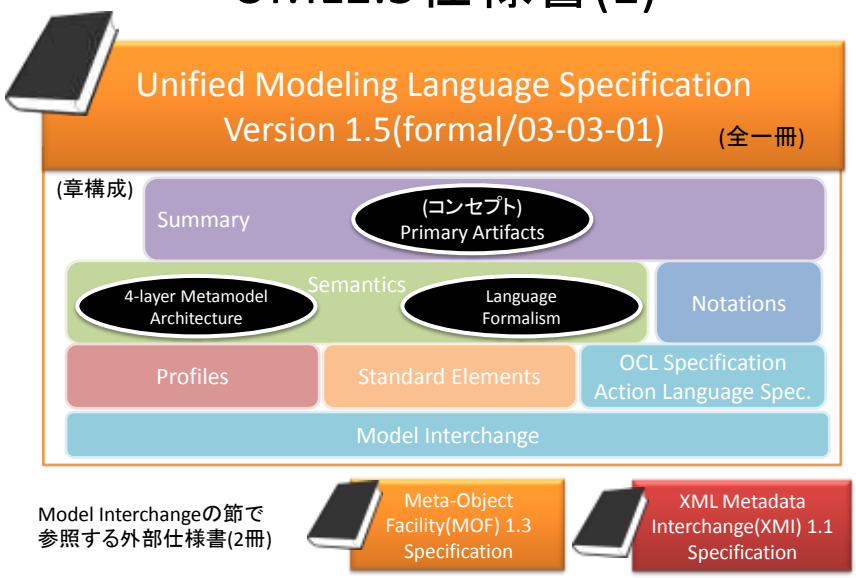
3.1 UML1.5仕様書の構造

- この節の目標
  - UML1.x系の最新バージョン1.5を題材に、仕様書の読み方の基本と、定義されているものについての大まかな知識を得る。
    - 仕様書の冊子・章構成
    - 内容を理解するためのコンセプト
    - 分類方法
    - パッケージとメタクラス



UML1.5

UML1.5仕様書(1)



UML1.5

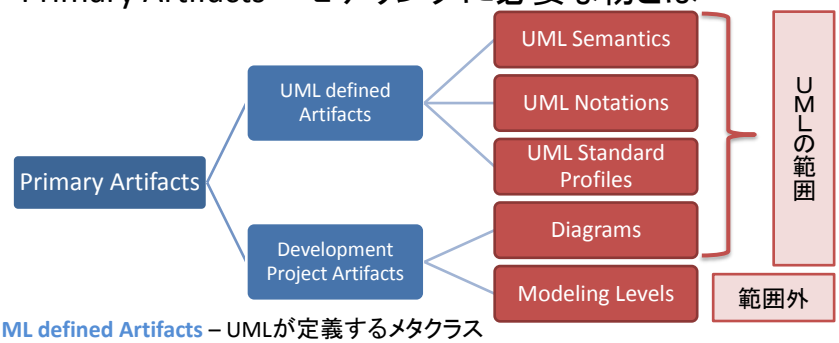
UML1.5仕様書(2)

- UML1.5仕様書は**1冊**から成る。
  - UMLモデルを他形式に変換するための「Model Interchange」の章では、「MOF1.3」と「XMI 1.1」仕様書を参照している。
  - 紙の書籍よりPDFなどの電子ファイルが便利
- 背景となる考え方として、**「Primary Artifacts」「4-layer Metamodel Architecture」「Language Formalism」**の3つのコンセプトがある。
  - これらのコンセプトは、仕様書の内容を理解する強力な助けとなる。
  - 「Primary Artifacts」は、何を定義するかについての考え方を述べている。
  - 「4-layer Metamodel Architecture」では、モデルを4つの階層に分ける考え方を説明している。
  - 「Language Formalism」では、言語を定義するための3つの観点について説明している。

UML1.5

Primary Artifacts(1)

- Primary Artifacts - モデリングに必要な物とは



UML defined Artifacts – UMLが定義するメタクラス
- UML仕様書の「Semantics」で定義する物とその記法、拡張方法

Development Project Artifacts - メタクラスの組み合わせで意味が発生するもの
- **Diagrams**: ユースケース図、クラス図、振る舞い図、実装図など図の種類
- **Modeling Levels**: 開発プロセスの各段階が要求するモデリングの内容
  - UMLで定義することの範囲外で、開発方法論によって異なる

UML1.5

Primary Artifacts(2)

	(UML Summary)	定義の背景となっているコンセプト、仕様書の章節構成を説明する。一度は通読したい部分。
Primary Artifacts	UML Semantics	モデリングに使う要素(クラスや関連など)の意味を定義する。必要に応じて参照する。
	UML Notations	同要素の書き方を定義する。必要に応じて参照する。
	UML Standard Profiles	同要素に独自の意味を追加するための拡張機能を定義する。必要に応じて参照する。
	Diagrams	同要素を使って作成する図の標準的な種類を定義する。Notationと合わせて解説されている。
	(その他仕様)	OCLなど、UML仕様の定義に使用しているその他の仕様を定義する。必要に応じて参照する。

Primary Artifactに「UML Summary」と「その他仕様」を加えたこれら6つは、UMLのバージョンによらず共通して仕様書に含まれている。

UML1.5

Primary Artifacts(3)

Unified Modeling Language

メタクラス

- メタモデル層のクラス(メタクラス)定義
- メタクラスの意味、ルール、記法、プロファイル
- メタクラスを分類するパッケージ

図

- クラス図
- オブジェクト図
- ユースケース図
- シーケンス図
- コラボレーション図
- ステートチャート図
- アクティビティ図
- コンポーネント図
- 配置図

モデリングレベル

- ビジネスモデリング
- 要求管理
- 分析設計
- プログラミング
- テスト

主旨

- 開発方法論のプロセスで要求されるモデリングレベルに柔軟に対応する。
- 標準的な図の種類を定義する。
- 図で使用できるメタクラスを定義し、意味、ルール、記法を統一する。
- メタクラスにユーザ独自の意味を追加する拡張を可能にする。

UML1.5

4-layer Metamodel Architecture

- 4つのメタモデル階層の中で、UMLは「メタモデル層」に属する

  (参考)
  - MOFはメタメタモデル層に属する
  - OMGの他のモデリング仕様であるCWM、BPMNなどもメタモデル層に属する

層	例	説明
メタメタモデル	Metaclass	このアーキテクチャの基礎となる層。メタモデルを作成するために使うもの。
メタモデル	Class	メタメタモデルのインスタンス。モデルを定義するために使うもの。
モデル	Person	メタモデルのインスタンス。モデリングしたもの。
ユーザーオブジェクト	John:Person	モデルをインスタンス化したもの。現実と一致する層。

この資料では以降、UMLで定義する要素を指す時には**「メタクラス」**という用語を使用する。

UML1.5

Language Formalism

- 自然言語に倣って、メタクラスの定義を3つの視点から行う
  - 言語定義としての正確性、完全性の向上を図る
- 各視点を仕様書の見出しとする
- 定義にはUML図、OCL式、英語を組み合わせて使用する

視点	説明	仕様書の見出し
Syntax	構文。メタクラスの意味とメタクラスそのものの構造についての説明。	Abstract Syntax
Static Semantics	静的意味。メタクラス間の静的な構造がもたらす意味と、使用上のルールについて。	Well-Formedness Rules
Dynamic Semantics	動的意味。メタクラス間の静的な構造とルールの組み合わせによって生じる意味の説明。	Semantics

あるメタクラスの定義をUML仕様書で調べる時に、
どの見出しの内容を参照すべきかのガイドとなる。

UML1.5

メタクラスの分類方法

分類1：メタクラスの意味上の役割で分類する

→ 仕様書第2章「UML Semantics」の目次に利用されている。

意味上の役割	説明	メタクラスの例
General Mechanisms	モデルの構造を表現する	Model, Package
Foundation	モデルの静的構造を表現する	Namespace, Class, Relationship, Stereotype, Expression
Behavioral Elements	モデルの動的な構造を表現する	Object, Message, UseCase, State
Actions	モデルの動作を表現する(UML1.5で初出)	Pin, ControlFlow, JumpAction, InvocationAction

分類2：メタクラスを使用する図(クラス図など)ごとに分類する

→ 仕様書第3章「UML Notification Guide」の目次に利用されている

単なる目次上の分類とネームスペースとしてのパッケージの差異はあいまい。

UML1.5

メタクラス一覧(1)

- **General Mechanisms**
  - Model Managementパッケージ
    - ElementImport, Package, Model, Subsystem
- **Foundation**
  - Coreパッケージ
    - **Backbone**: Element, ModelElement, ElementOwnership, Feature, Namespace, GeneralableElement, Palameter, Constraint, Classifier, StructuralFeature, BehavialFeature, Attribute, Operation, Method
    - **Relationships**: Relationship, Row, Generalization, Association, AssociationEnd, AssociationClass
    - **Dependencies**: Dependency, Binding, Abstraction, Usage, Permission
    - **Classifiers**: Class, Interface, Node, Component, Artifacts, DataType, Enumeration, Primitive
    - **Auxiliary elements**: TemplatePalameter, TemplateArgument, PresentationElement, Comment
  - Extension Mechanismsパッケージ
    - Constraint, Stereotype, TaggedValue, TagDefinition
  - Data Typesパッケージ
    - **Main**: Integer, UnlimitedInteger, String, Geometry, LocationReference, Mapping, MultiplicityRange, Name , <<enumeration>>AggregationKind, Boolean, CallConcurrencyKind, ChangableKind, OrderingKind, ParameterDirectionKind, PseudostateKind, ScopeKind, VisibilityKind
    - **Expressions**: Expression, ArgListsExpression, BooleanExpression, MappingExpression, ProcedureEdxpression, TimeExpression, TypeExpression

UML1.5

メタクラス一覧(2)

- **Behavioral Elements**
  - Common Behaviorパッケージ
    - **Signals**: Signal, Exception, Reception
    - **Procedure**: Procedure
    - **Instances:** AttributeLink, Stimulus, Procedure, Instance, DataValue, SubsystemInstance , ComponentInstance, NodeInstance, Object
    - **Links**: Link, LinkEnd, Stimulus, LinkObject
  - Collaborationパッケージ
    - **Roles**: Collaboration, AssociationRole, AssociationEndRole, ClassifierRole
    - **Interactions**: Interaction, Message
    - **Instances**: InteractionInstanceSet, CollaborationInstanceSet
  - Use Casesパッケージ
    - UseCaseInstance, Actor, UseCase, Include, Extend, ExtensionPoint
  - State Mahinesパッケージ
    - **Main**: StateMachine, Guard, StateVertex, Transition, Psudostate, SynchState, SubState, State, Transition, CompositeState, SimpleState, FinalState, SubmachineState
    - **Events**: Event, SignalEvent, CallEvent, TimeEvent, ChangeEvent
  - AcivityGraphsパッケージ
    - ActivityGraph, Partition, ActionState, ObjectFlowState, ClassifierInState, CallState, SubactivityState, Transition

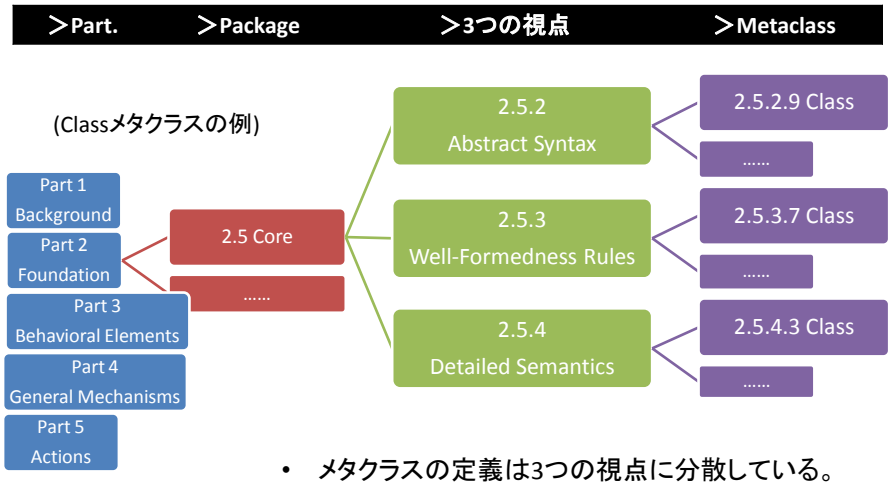
UML1.5

メタクラス一覧(3)

- **Actions**
  - Actionパッケージ
    - **Action Foundation**: Pin, ControlFlow, Action, PrimitiveAction, OutputPin, InputPin, DataFlow, Procedure
    - **Composite Actions**: Clause, GroupAction, Variable, ConditionalAction, LoopAction
    - **Read and Write Actions**: CreateObjectAction, DestroyObjectAction, ReclassifyObjectAction, ReadIsClassifiedObjectAction, AttributeAction, ReadAttributeAction, WriteAttrivuteAction, ClearAttributeAction, AddAttributeValueAction, RemoveAttributeValueAction, LinkAction, LinkEndData, QualifierValue, ReadLinkAction, ReadLinkObjectEndAction, ReadLinkObjectQualifierAction, WriteLinkAction, CreateLinkAction, LinkEndCreatiionData, CreateLinkObjectAction, DestroyLinkAction, ClearAssociationAction, VariableAction, ReadVariableAction, WriteVariableAction, AddVariableValueAction, RemoveVariableValueAction, ClearVariableAction, ReadExtentAction, ReadSelfAction, StartObjectStateMachineAction, CallProcedureAction
    - **Computation Actions**: ApplyFunctionAction, CodeAction, TestIdentityAction, LiteralValueAction, NullAction, PrimitiveFunction, ArgumentSpecification, UnmarshalAction, MarshalAction
    - **Collection Actions**: CollectionAction, MapAction, FilterAction, IterateAction, ReduceAction
    - **Messaging Actions**: ExplicitInvocationAction, InvocationAction, CallOperationAction, BroadcastSignalAction, SendSignalAction, SynchronousInvocationAction, AsynchronousInvocationAction
    - **Jump Actions**: JumpHandler, HandlerAction, JumpAction, BreakJump, ContinueJump

UML1.5

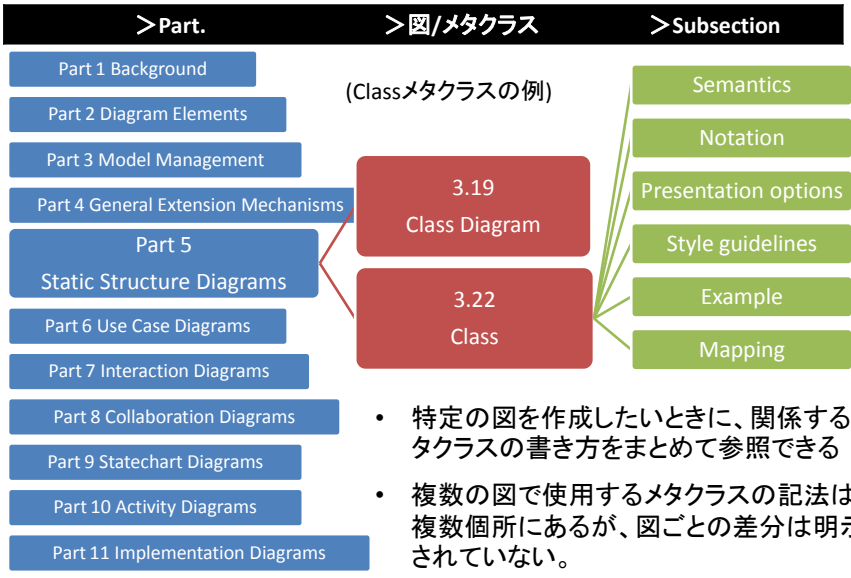
2章「UML Semantics」の構成



- メタクラスの定義は3つの視点に分散している。

UML1.5

3章「UML Notation Guide」の構成

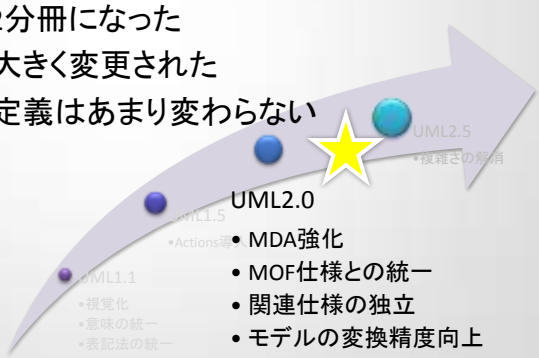


- 特定の図を作成したいときに、関係するメタクラスの書き方をまとめて参照できる
- 複数の図で使用するメタクラスの記法は複数個所にあるが、図ごとの差分は明示されていない。

UML2.0

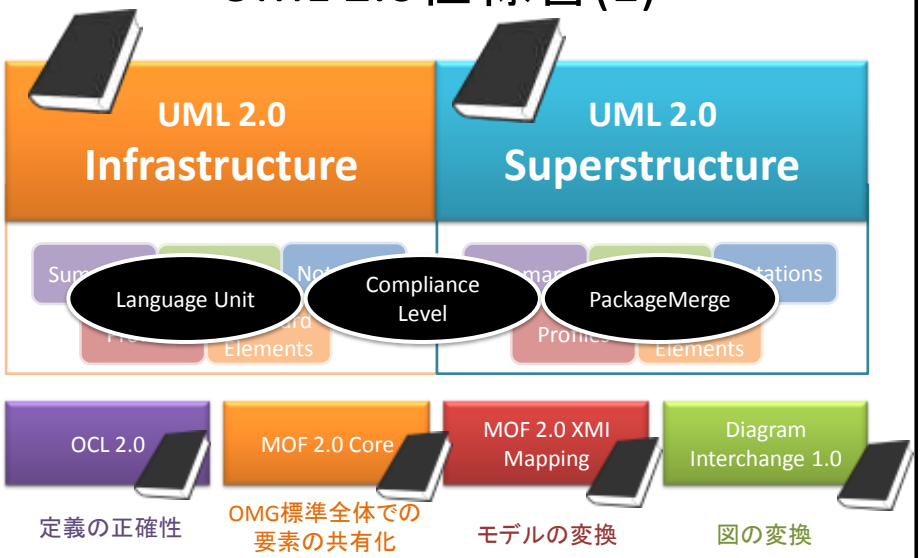
3.2 UML2.0仕様書の構造

- この節の目標
  - UML1.xからUML2.0で行われた変更点について知る。
    - UML定義が2分冊になった
    - 分類方法が大きく変更された
    - メタクラスの定義はあまり変わらない



UML2.0

UML 2.0仕様書(1)



UML2.0

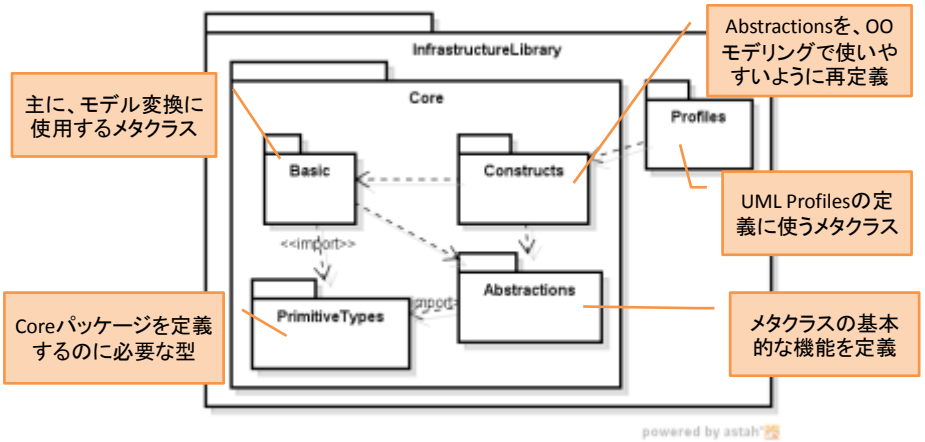
UML2.0仕様書(2)

- UML2.0仕様書は6冊の仕様書から成る。
  - **UML Infrastructure** (全1冊)
    - OMGのモデリング仕様(UML、MOF、CWM)の共通部分を抽出したもの
    - ツールメーカーやMDAユーザ向けで、一般的なユーザは読む必要なし。
  - **UML Superstructure** (全1冊)
    - UML2.0の知識を得たい人はSuperstructureだけを読めばよい。
  - **関連仕様を分冊化したもの**(全4冊)
    - OCL、MOF Core、MOF-XMI Mapping、DIの4冊。
    - 図の変換についての仕様が新規追加された。
- 2つの分類コンセプト**「Language Unit」「Compliance Level」**が追加された。
  - 「Language Unit」は、密接に関連するメタクラスをグループ化したもの。ユニットにはパッケージが属し、その中にメタクラスが属する。
  - 「Compliance Level」は、Language UnitにL0-L3のレベルを定義し、UML利用者とUMLツールにUML仕様を利用する基準を示すもの。
- **「PackageMerge」**機能が追加された。
  - あるメタクラスの定義をパッケージに分けてインクリメンタルに定義するために導入された。

UML2.0

「UML Infrastructure」の内容

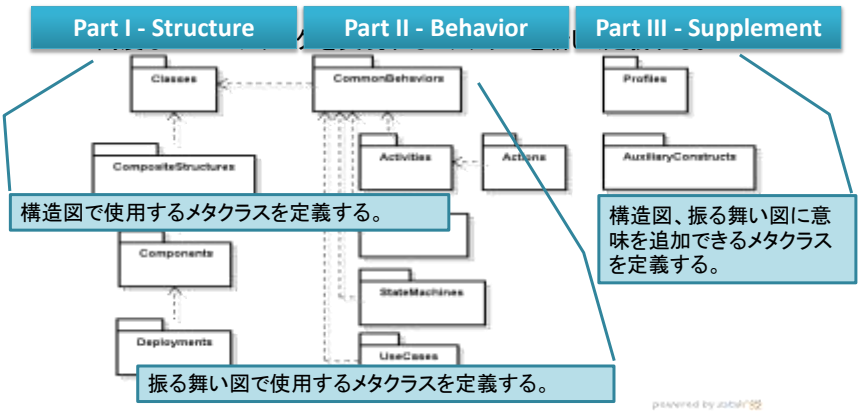
- モデリングの**基盤(Infrastructure)**となるメタクラスを定義する。
- MOF、UML Superstructureの**共通部分**を抽出して分類する。



UML2.0

「UML Superstructure」の内容

- Infrastructureで定義されたメタクラスを**再利用**する。
- **UMLモデリングで使用する**目的で、メタクラスの定義に詳細を追加する。



UML2.0

Language Unit

- 密接に関係する**モデリングコンセプト**によって、パッケージをグループ化する

Infrastructureのユニット

- Basic
- Constructs

Superstructureのユニット

- Actions
- Activities
- Classes
- Components
- Deployments
- General Behavior
- Information Flows
- Interactions
- Models
- Profiles
- State Machines
- Structures
- Templates
- UseCases

- ユニットは、基礎的なコンセプトからより高いモデリング能力を要するコンセプトへと**インクリメンタル**に分割されている。
- ユーザは自分のモデリングに必要な**ユニットごとに学習**すればよく、UML仕様全体を知る必要はない。

UML2.0

Compliance Level

- ユニットごとにサポートするパッケージを選択し、「UML」ネームスペースに統合する。
- ツールがUMLのどの部分をカバーするかを示すことができる。

レベル	説明
L0	InfrastructureのBasicユニットに属するパッケージをすべて統合する。Superstructureでは Kernelパッケージに該当する。データ交換可能な最小モデルをサポートする。
L1	L0+InfrastructureのConstructsユニットに属するパッケージとSuperstructureの10個のパッケージを統合する。クラス図、ユースケース図、シーケンス図、コミュニケーション図、アクティビティ図をサポートする。
L2	L1+Superstructureの15個のパッケージを統合する。コンポジット構造図、コンポーネント図、配置図、ステートマシン図、プロファイルをサポートする。
L3	L2+Superstructureの14個のパッケージを統合する。UMLのすべての定義をサポートする。

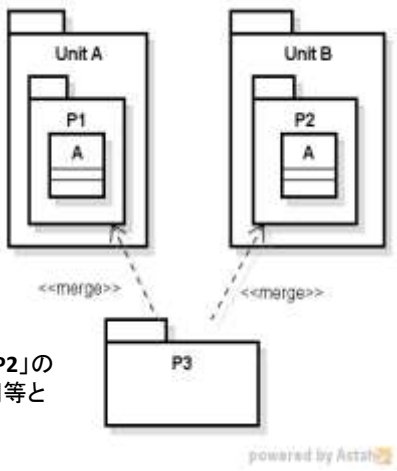
UML2.0

PackageMerge

- UML2.0で新規追加された機能。
- Compliance Levelごとにマージして1つのメタクラスとして利用できる。
- L3のUML::Class は、
  - Kernel
  - StructuredClasses
  - Communications
  - Profiles

  パッケージで定義されたすべてのClassメタクラスをマージしたメタクラスである。

  右図で、パッケージ「**P3**」には、「**P1**」と「**P2**」の定義をマージした「**A**」が存在するのと同等となる。



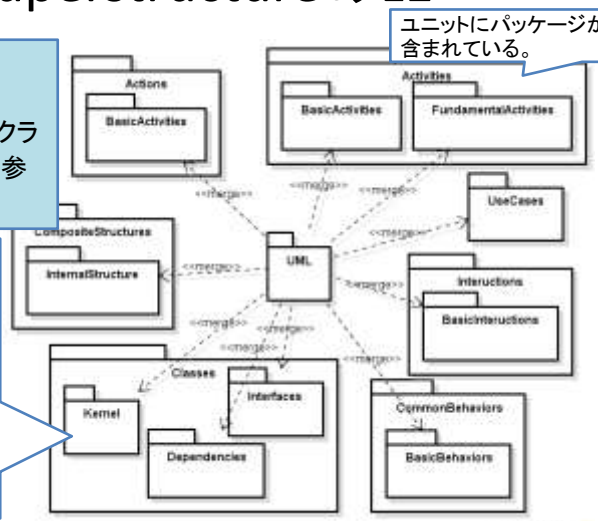
UML2.0

例：SuperstructureのL1

つまり、Superstructureの Compliance Level L1では、 Classes::Kernel::Class メタクラスを、**UML::Class** として参照できる。

SuperstructureのClassesユニットの**Kernel**パッケージは、InfrastructureLibrary::Coreパッケージの以下のサブパッケージを**PackageMerge**している。

- ✓Abstractions::Instances
- ✓Abstractions::MultiplicityExpressions
- ✓Abstractions::Literals
- ✓Abstractions::Generalizations
- ✓Constructs

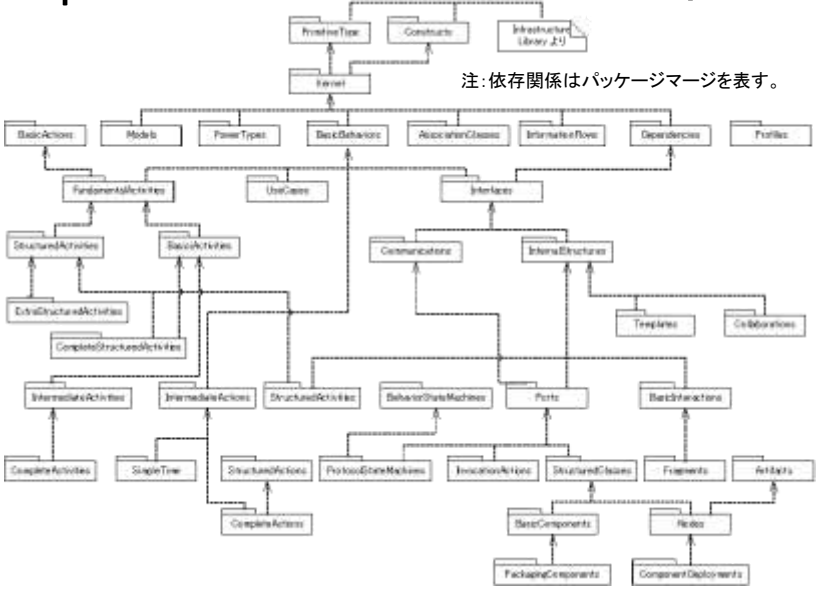


図：SuperstructureのL1レベルに含まれるパッケージとパッケージが属するユニット

UML2.0

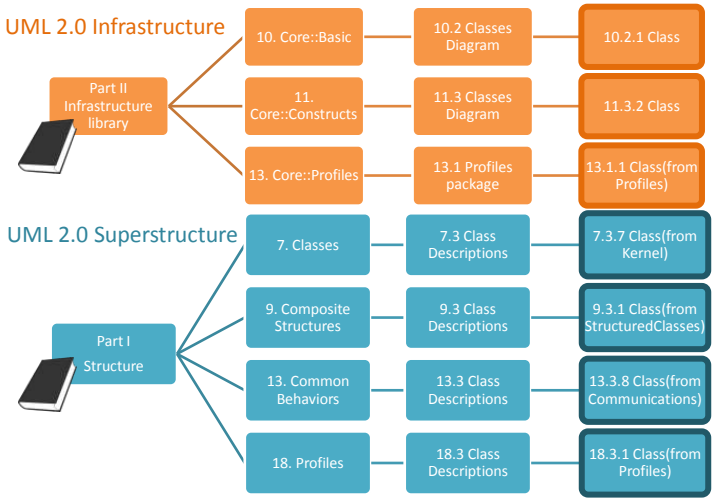
Superstructureパッケージ全体図

注：依存関係はパッケージマージを表す。



UML2.0

仕様書の構成



(Classメタクラスの例)

- Classメタクラスの定義は全**7ヶ所**に分散している。
- これらの定義をPackageMergeで一つにする。

Compliance Levelで範囲を明確にしたために、かえって仕様書の記述が分散した。

読みにくい

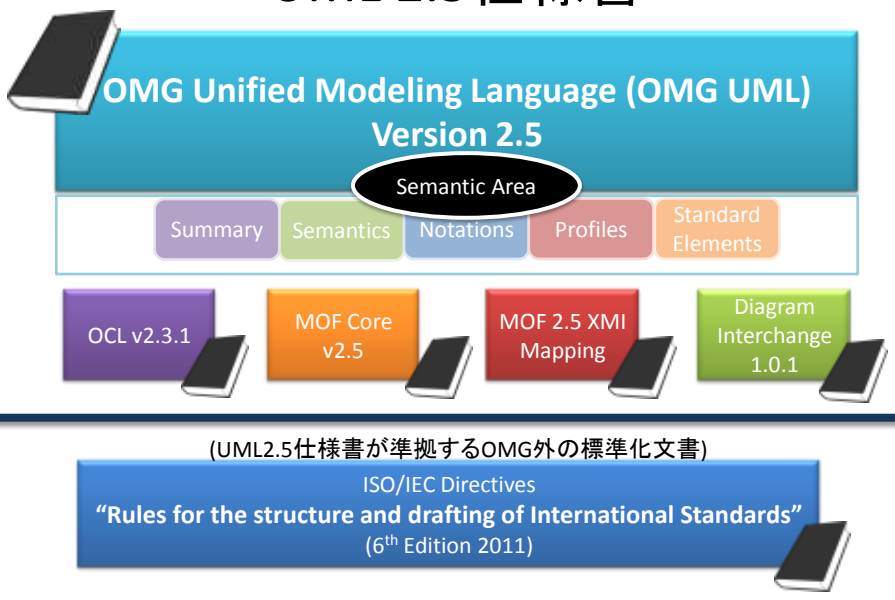
UML2.5

3.3 UML2.5での変更点

- この節の目標
  - UML2.0からUML2.5で行われた変更について知る。
    - UML定義が1冊になった
    - メタクラスの定義と記法は変わっていない
    - メタモデルの分類と仕様書の構成を一致させた
    - 不一致やあいまいさを改善した

UML2.5

UML 2.5仕様書



(UML2.5仕様書が準拠するOMG外の標準化文書)

ISO/IEC Directives
"Rules for the structure and drafting of International Standards"
(6th Edition 2011)

UML2.5

仕様書の改善

1冊に	• 「UML Infrastructure」はUML仕様から外れる。 • すべてのメタクラスは、引き続きトップレベルのUMLパッケージから直接参照できる。
PackageMerge 不使用	• それぞれのメタクラスを1つのパッケージだけで定義する。
仕様書の構造	• 読みやすさの向上のため、可能な限り前方参照しないようにする。
一覧追加	• すべての節にプロパティとプロパティに関連するメタクラスすべての一覧の項を追加した。 • 相互参照を容易にするため、ハイパーリンクを設定した。
Compliance Level 廃止	• UMLツールが実装するUMLのサブセット(メタモデル、記法、意味定義)はベンダーが明確にする。

～UML2.5仕様書「6.1 Specification Simplification」より引用～

UML2.5

補足：仕様書の改善

- 「UML Infrastructure」はどうなったのか？
  - UML2.5では「Meta Object Facility (MOF) Core, v2.5 」(OMG Specification ptc/2013-08-20)になる予定。
  - 「MOF v2.4.1」(最新)ではUML2.4.1の「Infrastructure」を再利用している。
  - UML2.4.1の「Infrastructure」はISO19505-1で規格化されているが、2.5はその兆しもない。
- PackageMerge不使用
  - L0-L3の分類をやめたため、メタクラスの定義をパッケージに分割する必要がなくなった。よってマージする必要もなくなった。
  - PackageMergeの機能自体はUML2.5にも存在している。
- 一覧追加
  - 「Classifier Descriptions」と「Association Descriptions」という項を節の下位に追加した。
  - このドキュメントは、機械可読のメタモデル定義から自動生成されている。

UML2.5

メタモデル定義の改善

(メタモデル自身は変更されていない)

一貫性	• モデルの構造と仕様書の構造を一致させた。 • パッケージ名をそのまま仕様書の節タイトルにした。
正確性	• OCL制約を積極的に使ってより正確に定義した。 • 関連と関連端の名称が両義性を避けるように一部変更された。
デフォルト値	• デフォルト値がある場合の多重度を0にした。
XMI変換	• LoopNode::loopVariableをコンポジットに変更した。 • NamedElement::clientDependencyを誘導可能にした。
集合	• {orderd}制約をちゃんと書いた。

～UML2.5仕様書「6.1 Specification Simplification」より引用～

UML2.5

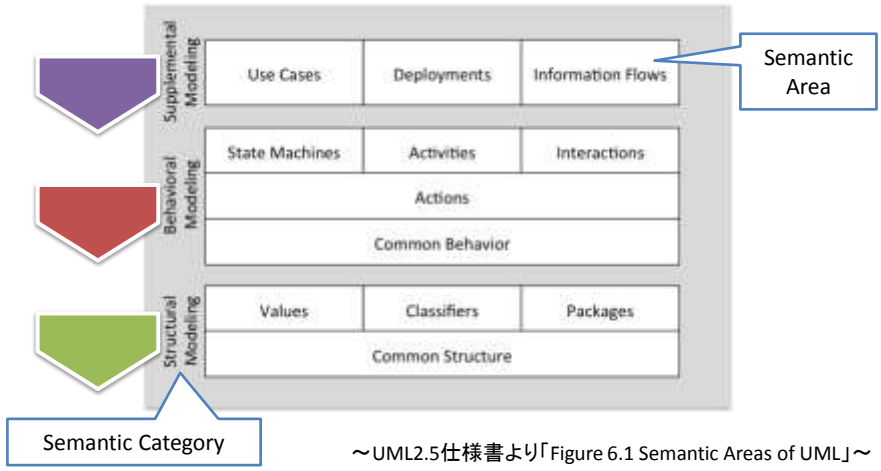
補足:メタモデルの改善

- 一貫性
  →「仕様書とモデル構造」「仕様書目次」スライド参照。
- 正確性
  - OCLの定義も2.2から2.3.1に更新された。
    - OclValid値/OclInvalid値の追加
    - UnlimitedNatural型(多重度の"*"に該当する)の追加
    - オブジェクトの型の違い(単一か集合か)による定義の違いの明文化
- XMI変換
  - 2.4.1以降、データ交換性を強化し、実際の使用に耐えるよう改善がなされた。
  - 2.5の目的の一つはXMIによる厳密な表現をしやすくするためのあいまいさの解消と全体的な仕様とメタモデルの構造の対応関係のシンプル化だと思われる。

UML2.5

仕様書とモデル構造

- モデル構造の基本分類



～UML2.5仕様書より「Figure 6.1 Semantic Areas of UML」～

UML2.5

仕様書目次

モデル構造と目次構成が一致している！

色分けが
Semantic Category

1. Scope
2. Conformance
3. Normative References
4. Terms and Definitions
5. Notational Conventions
6. Additional Information
7. Common Structure
8. Values
9. Classification
10. Simple Classifiers
11. Structured Classifiers
12. Packages
13. Common Behavior
14. State Machines
15. Activities
16. Actions
17. Interactions
18. UseCases
19. Deployments
20. InformationFlows
21. Primitive Types
22. Standard Profile

Annex A: Diagrams

Annex B: UML Diagram Interchange

Annex C: Keywords

Annex D: Tabular Notation for Sequence Diagrams Examples

Annex E: XMI Serialization and Schema

各節がSemantic Area

UML2.5

節の下位分類

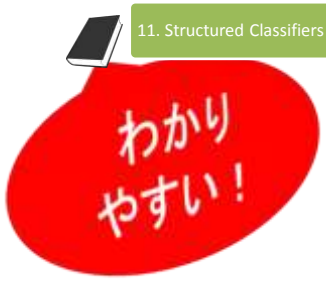
- Common Structure
  - Root
  - Templates
  - Namespaces
  - Types and Multiplicity
  - Constraints
  - Dependencies
- Values
  - Literals
  - Expressions
  - Time
  - Intervals
- Classification
  - Classifiers
  - Classifier Templates
  - Features
  - Properties
  - Operations
  - Generalization Sets
  - Instances
- Simple Classifiers
  - Data Types
  - Signals
  - Interfaces
- Structured Classifiers
  - Structured Classifiers
  - Encapsulated Classifiers
  - Classes
  - Associations
  - Components
  - Collaborations
- Packages
  - Packages
  - Profiles
- Common Behavior
  - Behaviors
  - Events
- StateMachines
  - Behavior StateMachines
  - StateMachine Redefinition
  - ProtocolStateMachines
- Activities
  - Activities
  - Control Nodes
  - Object Nodes
  - Executable Nodes
  - Activity Groups
- Actions
  - Actions
  - Invocation Actions
  - Object Actions
  - Link End Data
  - Link Actions
  - Link Object Actions
  - Structural Feature Actions
  - Variable Actions
  - Accept Event Actions
  - Structured Actions
  - Expansion Regions
  - Other Actions
- Interactions
  - Interactions
  - Lifelines
  - Messages
  - Occurrences
  - Fragments
  - Interaction Uses
  - Sequence Diagrams
  - Communication Diagrams
  - Interaction Overview Diagrams
  - Timing Diagrams
- UseCases
  - Use Cases
- Deployments
  - Deployments
  - Artifacts
  - Nodes
- InformationFlows
  - Information Flows
- Primitive Types
- Standard Profile

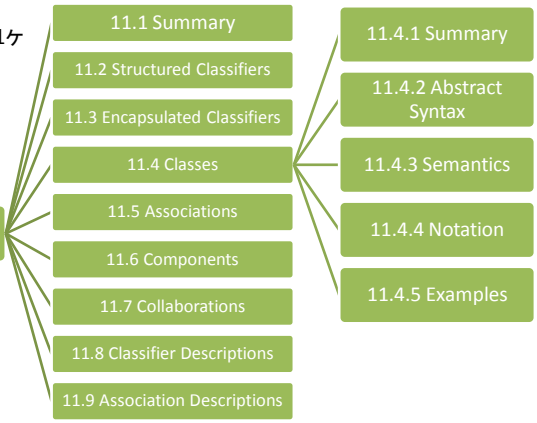
UML2.5

UML2.5仕様書の構成

(Classメタクラスの例)

- メタクラスが1つのパッケージに属するようになった。
- 構文、ルール、記法の解説が1ヶ所にまとまった。
- 仕様のあいまいさが低減した。
- 定義の一覧性が増した。
- サンプルが充実した。





4. その他の調査事項

Q1. シーケンス図のシーケンス番号について

- UML 2.5仕様書(ptc/2013-09-05) にシーケンス番号を持つ図の例があるが、UML2.5で復活したのか？

Q2. Interfaceメタモデルの属性と関連について

- UML1xではInterfaceは「属性は持たない」となっていたが、UML2xでは属性を持てる。
- UML1xではInterfaceは「関連は持たない」となっていたが、UML2xでは関連を持てる。

Q1. シーケンス図のシーケンス番号

- シーケンス番号を持つ図(Figure 17.10)は、
  - 図はUML2.5仕様書で初出で以前のバージョンには含まれていない。
  - 図はGeneralOrdering(中間に矢じりを持つ点線)の記法例である。
  - メタモデル定義にシーケンス番号を持つものはない。
  - 「17.9 Communication Diagrams」にのみSequence expressionの項がある。
  - Annex Bの「B.4.5 Interaction Tables」にUMLInteractionTableLabelKind.sequenceNumber値の定義があるが、これはシーケンス図を表形式に落とす場合、列の種類を表す。(表形式についてはAnnex Dにサンプルがある)

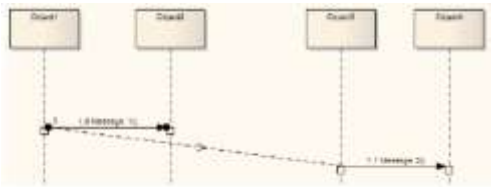


Figure 17.10 Example showing general Ordering in a sequence diagram

A1. 図の間違いと思われる。UMLツールでは描画可能なものもあるが、UML1.xとの互換性のためではないか。

Q2. Interfaceの属性と関連

A2. UML2.0で仕様が変更されました。

- 「属性を持つ」ことと「関連端を持つ」ことは同義です。
- InterfaceメタクラスはPropertyをownedAttributeとして持つことができるようになっています。
- そもそも、CORBA IDLでinterfaceにはattributeがありましたから、UML 1.xでAttributeを持てないとしたのはこれと矛盾していたのではないでしょうか？